Preliminary Draft

NIST Special Publication 800-XX
Version 0.1

Integrated Circuit Card for Personal Identity Verification

National Institute of Standards and Technology
Technology Administration
U.S. Department of Commerce

INFORMATION SECURITY

Computer Security Division
Information Technology Laboratory
National Institute of Standards and Technology
Gaithersburg, MD 20899-8930

February 2005

Working Paper

REPORTS ON COMPUTER SYSTEMS TECHNOLOGY

Authority

This document has been developed by the National Institute of Standards and Technology (NIST) in furtherance of its statutory responsibilities under the Federal Information Security Management Act (FISMA) of 2002, Public Law 107-347.

NIST is responsible for developing standards and guidelines, including minimum requirements, for providing adequate information security for all agency operations and assets, but such standards and guidelines shall not apply to national security systems. This recommendation is consistent with the requirements of the Office of Management and Budget (OMB) Circular A-130, Section 8b(3), Securing Agency Information Systems, as analyzed in A-130, Appendix IV: Analysis of Key Sections. Supplemental information is provided A-130, Appendix III.

This recommendation has been prepared for use by federal agencies. It may be used by nongovernmental organizations on a voluntary basis and is not subject to copyright. (Attribution would be appreciated by NIST.)

Nothing in this document should be taken to contradict standards and guidelines made mandatory and binding on federal agencies by the Secretary of Commerce under statutory authority. Nor should this recommendation be interpreted as altering or superseding the existing authorities of the Secretary of Commerce, Director of the OMB, or any other Federal official.

National Institute of Standards and Technology, Draft Special Publication 800-*XX*
Natl. Inst. Stand. Technol. Spec. Publ. 800-*XX*, 73 pages (*February2005*)

Acknowledgements

The author wishes to thank his colleagues who reviewed drafts of this document and contributed to its development. The author also gratefully acknowledges and appreciates the many comments from the public and private sectors whose thoughtful and constructive comments improved the quality and usefulness of this publication.

Abstract

An integrated circuit card is a portable and tamper-resistant data store that includes processing capability. The data store is implemented as a file system with directories containing files and subdirectories. The processing capabilities are implemented as built-in platform commands and add-on applications. A client-application program using an integrated circuit card sends card commands to the card via the card command interface by means of placing calls on the high-level, task-oriented client-application programming interface. The card commands themselves are executed inside the integrated circuit card. The results of executing the commands are returned to client-application program by the client-application programming interface. This chapter of Special Publication 800-*XX* (800-*XX*) describes both a client-application programming interface and built-in card platform command set for a particular integrated circuit card, the Federal Personal Identity Verification (PIV) card. The Special Publication is provided in sufficient technical detail that compliant application programs and compliant integrated circuit cards can be used interchangeably by any information processing system conforming to the standard.

Key words: Integrated circuit card, Card elements, Card applications, Security Architecture

[This page intentionally left blank.]

Table of Contents

1. INTRODUCTION

An integrated circuit card is a portable and tamper-resistant data store that includes processing capability. The data store is implemented as a file system with directories containing files and subdirectories. The processing capabilities are implemented as built-in platform commands and add-on applications.

A client-application program using an integrated circuit card sends card commands to the card via the card command interface by means of placing calls on the high-level, task-oriented client-application programming interface. The card commands themselves are executed inside the integrated circuit card. The results of executing the commands are returned to client-application program by the client-application programming interface.

This chapter of Special Publication 800-*XX* (800-*XX*)describes both a client-application programming interface and built-in card platform command set for a particular integrated circuit card, the Federal Personal Identity Verification (PIV) card.

The Special Publication is provided in sufficient technical detail that compliant application programs and compliant integrated circuit cards can be used interchangeably by any information processing system conforming to the standard.

SP 800-*XX* is organized as follows:

Section 2, Terms, Acronyms, and Notation, describes the vocabulary and textual representations used in the document in one place for easy reference.

Section 3, Concepts and Constructs, describes the model of computation of the PIV, the information processing concepts and the data constructs.

Section 4, Data Types and their Representations, provides the details of the data found on the client-application programming and card command interfaces.

Section 5, Client-Application Programming Interface, describes the application programming interface in programming language independent terms.

Section 6, Card Platform Command Interface, describes the card command interface.

Section 7, Common Data Model for Personal Identification Verification, describes specific data stored on the PIV.

Section 8, References, lists the documents on which this Special Publication is based.

2. TERMS, ACRONYMS, AND NOTATION

2.1 Terms

Application Identifier	A globally unique identifier for a loadable application as specified by ISO/IEC 7816-5.
Application Session	The period of time within a card session between when a card application is selected using its application identifier and another application is selected or the integrated circuit card is reset.
Card	An integrated circuit card.
Card Application	Set of data elements and associated card command implementations that can be selected using an application identifier (AID).
Card Manager	The distinguished card application present on every PIV integrated circuit card. The Card Manager is the currently selected application when the card is powered up or reset.
Card Reader	Synonym for interface device.
Card Session	The period of time between when a card is reset and either power is removed from the card or the card is reset again.
Client Application	A computer program running on a computer connected to an interface device containing an integrated circuit card and using the application programming interface described herein to access the capabilities of the integrated circuit card.
Data Element	An item of information seen at the card command interface for which are specified a name, a description of logical content, a format and a coding.
Entity	Any participant in an authentication exchange with the PIV; such a participant may be human or nonhuman.
Interface Device	The electronic device that provides the integrated circuit card with power and with which the integrated circuit card is in physical layer communication.
Principal	An entity whose credentials can be authenticated using reference data and authentication protocols on the PIV.

Reference Data	Data used to perform an authentication protocol for a specific principal. Examples are passwords, PINs, and cryptographic keys used for authentication.
Reset	A signal sent to the integrated circuit card that causes the card to delete all current state and reinitialize itself. A *warm reset* is affected by means of a reset signal sent to the card interface. A *cold reset* is affected by means of a power-on reset of the card.
Status Word	Two bytes that are returned by the integrated circuit card after the processing any command that signify the success of or errors encountered during said processing.
Template	A constructed BER-TLV data object containing specified data objects. Templates are used to transmit collections of data object pertaining to a particular context or purpose. The tag of the template identifies this context or purpose.

2.2 Acronyms

ADF	Application Dedicated File
AID	Application Identifier
BER	Basic Encoding Rules
CLA	Class (first) byte of a card command
CRT	Control Reference Template
DF	Dedicated File
DOT	Data Object Tag
EF	Elementary File
FCP	File Control Parameters
FDB	File Descriptor Byte
FID	File Identifier
ICC	Integrated Circuit Card
INS	Instruction byte of a card command
P1	First parameter of a card command
P2	Second parameter of a card command
PIV	Integrated Circuit Card for Identification
IFD	Interface Device
INS	Instruction (second) byte of a card command
LSB	Least Significant Bit
MF	Master File
MSB	Most Significant Bit
PIN	Personal Identification Number
PIV	Personal Identity Verification
RFU	Reserved for Future Use
SW1	First byte of a two-byte status word
SW2	Second byte of a two-byte status word

TF	Transparent File
TLV	Tag-Length-Value

2.3 Notation

The sixteen hexadecimal digits are denoted using the alphanumeric characters 0, 1, 2…, A, B, C, D, E, and F. A byte consists of two hexadecimal digits. Each byte is represented by bits b8 to b1, where b8 is the most significant bit (MSB) and b1 is the least significant bit (LSB). In each textual or graphic representation, the leftmost bit is the MSB. Sequences of bytes will be enclosed in be enclosed in apostrophes, for example, ‘2D’ and '3F 00'.

In the PIV, all bytes specified as reserved for further use (RFU) shall be set to '00' and all bits specified as RFU shall be set to 0.

All lengths are measured in number of bytes unless otherwise noted.

Data objects in templates are described as being Mandatory (M), optional (O) or conditional (C). In the case of conditional data objects, the conditions under which they are required are provided.

3. CONCEPTS AND CONSTRUCTS

The PIV client-application programming interface provides a high-level and programming-language-independent interface to the capabilities of the low-level PIV card command interface. The information processing concepts and data constructs on both interfaces are identical and may be referred to as PIV information processing concepts and data constructs without reference to a particular interface.

Theclient-application programming interface provides task-specific programmatic access to these concepts and constructs and the card command interface provides communication access to concepts and constructs.

The client-application programming interface is thought of as being at a higher level than the card command interface because access to a single entry point on the client-application programming interface may cause multiple card commands to traverse the card command interface. In other words, it may require many commands on the card interface to accomplish the task represented by the client-application entry point.

The client-application programming interface is a program execution, call/return style interface where as the card command interface is communication protocol, command/response style interface. Because of this difference the representation of the PIV concepts and constructs as bits and bytes on the client-application program interface is may be different from the representation of these same concepts and constructs on the card command interface.

3.1 Data Elements

In order to support both file-based and object-based views of data storage, this specification uses the term *data element* to refer generically to either view of data.

A *data element* is an item of information seen at the card command interface for which are specified a name, a description of logical content, a format and a coding.

In particular and in conformance with the ISO 7816 series of integrated circuit card standards, a data element is either a *transparent file* identified by a two-byte file identifier (FID) or a BER-TLV *data object* identified by a BER-TLV data object tag (DOT).

3.1.1 Data Content

The *content* of a data element is the sequence of bytes that are said to be *contained in* or to be the *value of* the data element. The number of bytes in the sequence is the *length* of the data contained in the data element and also the *size* of the data element.

The first byte in the sequence is regarded as being at *byte position* or *offset* zero in the content of the data element.

A transparent file may contain data objects. In this case the file descriptor byte of the transparent file indicates that the structure of the transparent file is “TLV structure for BER-TLV data objects.” We will refer to such a transparent file as a *BER-TLV file*. A transparent file that is not a BER-TLV file is called an *unstructured transparent file*.

A data object may contain other data objects. In this case the tag of the data object indicates that data object is a *constructed data object*. A data object that is not a constructed data object is a called a *primitive data object*.

3.1.2 Data Element Organization and Naming

Each unstructured transparent file and primitive data object is a leaf node of a rooted tree called a *file system*.

Interior nodes of a file system are BER-TLV files, constructed data objects and special nodes called *dedicated files*. A dedicated file is called a directory, a folder or a container in personal computing file systems. Each dedicated file in a file system is identified by a two-byte file identifier.

If a data element is directly connected to a dedicated file in a file system, the data element is said to be *contained in* the dedicated file.

Each PIV integrated circuit card contains one or more file systems. The root of each file system is associated with and thus named by an application identifier (AID) as defined by ISO/IEC 7816-5. The root of a file system identified by an AID is called an *application dedicated file* (ADF). The integrated circuit card may contain a distinguished root called the *master file* (MF).

Using the above organization, each data element on a PIV integrated circuit card is uniquely identified and hence named by a sequence consisting of 1) an AID (a root) followed by 2) a sequence of zero or more dedicated file FIDs, BER-TLV FIDs or constructed DOTs (the interior nodes), and terminated by 3) the FID of a unstructured transparent file or the DOT of a primitive data object.

The following are examples of fully-elaborated PIV integrated circuit card data element names. The symbol | denotes concatenation.

- Primitive data object within an application: AID | DOT
- File in a child dedicated file of the root of a file system: AID | FID | FID
- Data object within a constructed data object in an application: AID | DOT | DOT
- Data object in a BER-TLV file in a root dedicated file: AID | FID | DOT

On both the client-application programming interface and the card command interface, data elements are referred to relative to the current state of the PIV integrated circuit card (see 3.5 below) which includes a currently selected application and optionally a currently selected dedicated file and a currently selected data element.

The following are examples of data element names relative to the current state which is how they are used on the client-application programming and card command interfaces described in this document.

- Primitive data object within the currently selected application: DOT
- File contained in the currently selected dedicated file of the currently selected application: FID
- Data object in the currently selected BER-TLV transparent file in the currently selected dedicated file of the currently selected application: DOT

3.1.3 Currently Selected Dedicated File and Currently Selected Data Element

Within each file system on the PIV integrated circuit card there may be a distinguished dedicated file called the *currently selected dedicate file*. There may also be a distinguished data element that is contained in the currently selected dedicated file called the *currently selected data element*.

A dedicated file in a file system can become the currently selected dedicated file in the file system by means of the use of the SELECT card command (see below) or by means of the action of a card application. The same is true for a data element becoming the currently selected data element.

3.1.4 Adding and Deleting Data Elements

Data elements can be added to and deleted from a file system using the data management entry points on the client-application programming interface.

Adding a data element to a file system may entail extending the file system by adding one or more dedicated files in order to place the new data element in the proper place it the hierarchy.

3.2 Card Applications

Commands on the card command interface in addition to the PIV card platform commands may be provided by a *card application* on the PIV integrated circuit card. A card application is an executable program stored on the integrated circuit card that performs the processing defined by the commands it surfaces on the card command interface.

A card application may be placed in the PIV integrated circuit card during its manufacturing and personalization phases or it may be loaded onto the PIV integrated circuit card after the card has been issued to the cardholder and is in use. In the latter case the application is called a *loadable card application*.

It is not mandatory that a PIV integrated circuit card support the loading of card applications after the card has been issued and is in use. If a PIV integrated circuit card does support card application loading, the loading of the card application onto the PIV integrated circuit card shall be accomplished using the card content management commands described herein.

3.2.1 Card Manager Application

The *card manager application* is a card application that is present on every PIV integrated circuit card. The AID of the PIV card manger application is 'A0xxxxxxxxxxxxxxxx' and the root of the file system associated with the card manager is the master file.

The card manager surfaces all of the card platform commands on the card command interface.

3.2.2 Card Platform Commands

The card manager application surfaces all of the card commands described herein on the card command interface. These commands are built into the PIV integrated circuit card as the commands of the card manager application and hence are called the *card platform commands*.

A PIV integrated circuit card may surface additional commands on the card command interface. These are provided by card applications as described below and are called *card application commands*.

3.2.3 Currently Selected Card Application

Each card application is identified by a globally unique application identifier (AID) as defined by ISO/IEC 7816-5. A card application on a PIV integrated circuit card is activated by selecting the card application using the card applications AID. At most one card application shall be active on the PIV integrated circuit card at any time. If a card application is active, it is called the *currently selected card application*.

3.2.4 Card Application Data

The AID of a card application may be also associated with a file system on the PIV integrated circuit card.

In this case and unless otherwise noted in the documentation of the card application, when the card application is activated, the root of this file system becomes the currently selected dedicated file or currently selected data element depending on whether the root is a dedicated file or a data element respectively. If the root of the file system is a dedicated file, the activation of the application may also set one of the data elements in this dedicated file as the currently selected data element.

A dedicated file distinguished as the currently selected dedicated file by the activation of a card application is called the *default dedicated file* of the card application. A data element distinguished as the currently selected data element by the activation of a card application is called the *default data element* of the card application.

3.2.5 Adding and Deleting Card Applications

Card applications can be added to the integrated circuit card and existing card applications can be deleted using the application management entry points on the application programming interface and the card content management card commands.

3.2.6 Illustration of Card Application and Data Relationships

Figure 1 below illustrates the relationship between card applications and card file systems.

In this situation, Application A and Application B are sharing file system rooted at ADF #1. Application C shares distinguished file system rooted at MF with the card manager application. The card manager application also implements the data-only application rooted at ADF #2. Application D and Application E each have their own file systems rooted at ADF #3 and ADF #4 respectively.

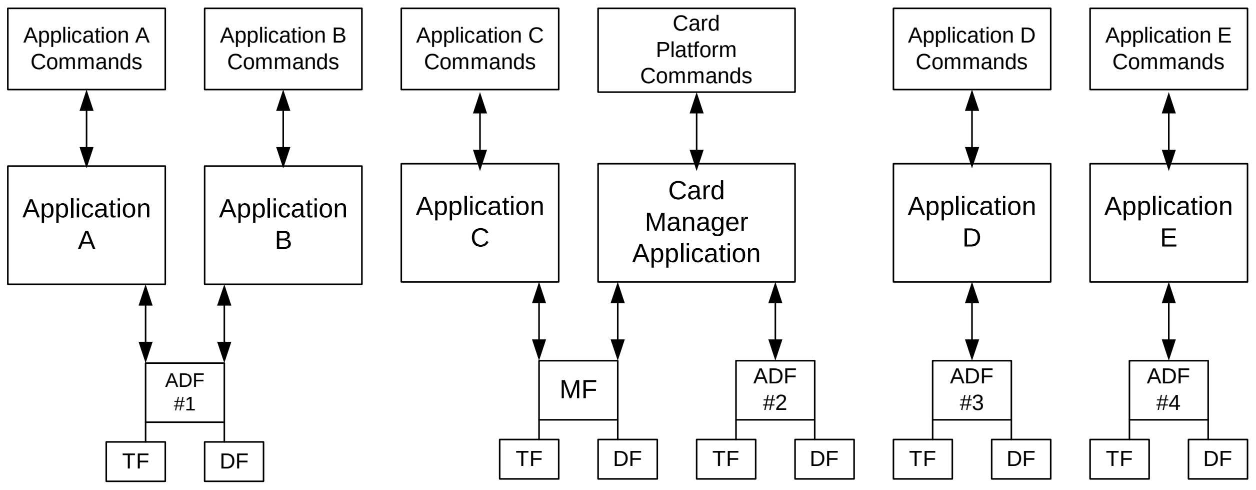

Figure 1: Illustration of Card Application and Data Relationships

One uses the card platform commands documented below to manage and use the data in the distinguished file system rooted at MF as well as the data-only application rooted at ADF #2. Application C surfaces its own card commands on the card command interface but uses data in the distinguished file system. All of the other applications surface their own card commands on the card command interface and store data in and use data from their own file systems.

3.3 Security Architecture

The security architecture of the PIV integrated circuit card provides concepts and technical machinery to control access to the data and processing capabilities of the PIV integrated circuit card.

In a nutshell, an access control rule is associated with each data element on the card that says who can perform which operations on the data element. The individuals mentioned in an access control must be authenticated by the card before the operations by the rule are allowed.

3.3.1 Principals

A *principal* is an entity whose identity can be authenticated using PIV data and algorithms stored on the PIV. Examples of principals are human beings, organization entities and information processing systems.

Each principal is uniquely identified on the PIV by a one-byte value called a *reference data identifier*. The reference data identifier identifies the reference data that is used to authenticate the principal. Reference data can be a password or a PIN in the case that the principal is a human being and it may be a cryptographic key in the case that the principal is an information processing system or an organizational entity.

3.3.2 Security Status

Associated with each principal is a Boolean variable called the *security status indicator* of the principal. The security status indicator of a principal is TRUE if the credentials of the principal are considered to be currently authenticated and FALSE otherwise.

All security status indicators are set to FALSE when the integrated circuit card is reset.

The successful execution of an authentication protocol sets the security status indicator of the principal whose credentials were verified by the protocol to TRUE.

A security status indicator is said to be a *global* security status indicator if it is not reset when the currently selected application changes from one application to another.

A security status indicator is said to be an *application* security status indicator if it is reset the currently selected application changes from one application to another. Every security status indicator is either a global security status indicator or an application security indicator status.

The term *global security status* refers to the set of all global security status indicators. The term *application security status* refers to the set of all application security status indicators.

The security status of a particular application, called the *application's security status*, is the set of all security status indicators of which the application can change the value. An application's security status can contain both global and application security status indicators.

3.3.3 Access Control Rules

An *access control rule* consists of an *access mode* and a *security condition*. The access mode is an action that can be performed on a data element. A security condition is a Boolean expression in security statuses.

According to an access control rule, the action of the access mode can be performed if and only if the security condition values to TRUE. If there is no security condition associated with the access mode then the access mode action may never be performed on the data element.

For example, if the access control rule

```
(READ, Cardholder OR Card Issuer)
```

were associated with a data element, then the data object could be read if and only if either the Cardholder or the Card Issuer had been authenticated. If the current security status does not indicated that one or the other of these is currently authenticated then the data element could not be read.

3.3.4 Current Security Environment

Associated with the currently selected application is a set of data objects called the *current security environment*. These data objects parametrize cryptographic operations performed by the card application.

When a card application is selected, the current security environment is set to a set of data objects defined by the application called the card application's *default security environment*.

Data objects in the current security environment can be created and changed on the card command interface using the MANAGE SECURITY OPERATION command.

3.4 Current State of the PIV Integrated Circuit Card

The contents of the *current state* of a PIV integrated circuit card are listed and described in the following table.

Table 3-1 – Current State of a PIV Integrated Circuit Card

State Name	**Always Defined**	**Comment**
Currently selected application	Yes	There is always a currently selected application; the default application is the Card Manager application
Currently selected file system	Yes	May be null in the unusual case that an application stores no persistent data.
Currently selected dedicated file	No	The root of a file system could be a constructed BER-TLV data object in which case.
Currently selected data element	No	Immediately after the selection of a dedicated file and after the deletion of the currently selected data element there is no currently selected data element
Global security status	Yes	Never null; always contains the cardholder
Application security status	Yes	May be null
Current security environment	No	Support of a current security environment by an application is optional.

3.4.1 Secured Data Exchange

Encryption, cryptographic checksums and digital signatures can be applied to data flowing in the communication connetion between a client-application and an integrated circuit card. On the client-application programming interface this is called a *secure channel*. On the card command interface this is called *secure messaging*. Secure channels are implemented at least in part using secure messaging.

3.4.2 Cryptographic Information Application

The *cryptographic information application* is a codified and structured on-card database of information about the cryptographic capabilities on the PIV integrated circuit card. The general cryptographic information application is described in the international standard ISO/IEC 7816-15. Described herein is the particular structure and contents of the cryptographic information application found on a PIV integrated circuit card.

The AID of the cryptographic information application on the PIV integrated circuit card is 'E8 28 BD 08 0F 00'. Selecting this AID sets the currently selected dedicated file to DF.CIA. The transparent files and dedicate files found in this ADF are described in Section 6.6 below.

3.5 Secure Messaging and Command Chaining Indication

Bit settings in the first byte, the class byte (CLA), of a card command are used to signal special processing behavior with regards to a command on the card command interface.

If bits 3 and 4 in the class byte are set to 1 then secure message processing has been applied to the command. If bits 3 and 4 are 0 then no secure message processing has been applied to the command. Secure messaging is described in detail in Section 6.5 below.

If bit 5 in the class byte is set to 1 then this command is not the last command in a sequence of commands called a *chain*. The semantics of a chain of commands is that all the commands in a chain must execute successfully or the state and contents of the integrated circuit card must be reset to the state and contents at the time the first command in the chain was received. In other words, the chain forms a transaction all of which or none of which shall be executed. Command chaining is described in detail in Section 6.6 below.

Table 3-2: Class Byte of a Card Command

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
0	0	0	0	-	-	0	0	Last or only command in a chain
0	0	0	1	-	-	0	0	Not the last command in a chain
0	0	0	-	0	0	0	0	No secure message processing applied to command
0	0	0	-	1	1	0	0	Secure message processing applied to the command

Depending on the functionality of an individual card command it may support one or the other or both or neither of these special processing behaviors. The supported behaviors for each card command are included in the description of the card command and summarized in table at the beginning of the card command interface section.

3.6 Card Communication

Communication with a FIPS 201 PIV is described in this Special Publication at the transport layer of the ISO OSI reference model. The physical, data link layers and network of the communication between the client-application programming interface and the card command interface including communication between the client application platform and the card reader are described in Section 6 of FIPS 201.

A card command consists of a 4-byte command header followed by the length of the command data field, the command data field, and terminated by the length of the expected response data field. A command response consists of a command response data field of 0 or more bytes followed by a two-byte status word.

4. DATA TYPES AND THEIR REPRESENTATIONS

This section provides a description of each data type found on the client-application programming and card command interfaces. Unless otherwise indicated the representation is the same on both interfaces.

The representation of 'sequence of' on the client-application programming interface is programming language binding dependent.

4.1 Access Control Rule

An access control rule, possibly null, is associated with every card application, every dedicated file, every transparent file, and every data object.

An access control rule describes the security conditions under which an operation may be performed on the entity with which it is associated. The semantics of a null access control rule is that no operation may ever be performed on the entity.

Access control rules are encoded in the expanded format described in ISO/IEC 7816-4.

A PIV integrated circuit card may optionally support access rule referencing as defined in ISO/IEC 7816-4.

4.2 Algorithm Identifier

An algorithm identifier is a one-byte identifier of a cryptographic algorithm together with a mode of operation and reference data length. For the match algorithm, the reference data length is the maximum length of a password or PIN. For the other algorithms, the reference data length is the length of a key.

Algorithm Identifier	Algorithm-Mode	Reference Data Length (Bits)	Padding
	Password/PIN Match	64	Null
	Triple DES-ECB	128	Null
	RSA	1024	PKCS #1
	Etc.		

4.3 Application Identifier

An application identifier (AID) is sequence of from 5- to 16-bytes as described in ISO/IEC 7816-5.

4.4 Application Properties

Table 4-1 – Data Objects in a Card Application Property Template

Description	Tag	M/O
Application Identifier of application	4F’	M
Application label	‘50’	O
Uniform resource locator	‘5F50’	O

4.5 Authenticator

Table 4-2 – Data Objects in an Authenticator Template

Description	Tag	M/O
Access mode	‘80’	M
Authentication data	‘81’	M
Reference data identifier	‘83’	M

4.6 Boolean

A Boolean is a byte with the interpretation that the value '00' represents FALSE and all other values represent TRUE.

4.7 Byte

A byte is a sequence of 8 bits.

4.8 Connection Description

Table 4-3 – Data Objects in a Connection Description Template

Description	Tag	Comment	M/O
Interface device identifier – PC/SC	‘80’	Card reader name	O
Interface device identifier – SCP	‘81’	Card reader identifier	O
Network node identifier - Internet	‘84’	Internet domain name or IP address	O
Network node identifier - Telephony	85’	ISDN dialling number string	O

4.9 Data Object

A data object is sequence of bytes that has been BER-TLV encoded according to ISO/IEC 8825-1:2002.

4.10 Data Element Name

A data element name is either a two-byte file identifier or a BER-TLV tag.

4.11 Data Element Properties

Table 4-4 – Data Objects in a Transparent File Property Template

Description	Tag	Comment	M/O
Number of data bytes in the file	'80'		M
File descriptor byte	'82'		M
File identifier	'83'		M
Security attribute in expanded format	'AB'		M

Table 4-5 – Data Objects in a Data Object Property Template

Description	Tag	Comment	M/O
Number of data bytes in the value	'80'		M
Tag	'83'		M
Security attribute in expanded format	'AB'		M

Table 4-6 – Data Objects a Dedicated File Property Template

Description	Tag	Comment	M/O
File descriptor byte	'82'		M
File identifier	'83'		M
Security attribute in expanded format	'AB'		M

4.12 Handle

A handle is an opaque sequence of bytes that identifies a transient resource such as a connection between a client-application and a particular integrated circuit card.

4.13 Reference Data Identifier

A reference data identifier is a one-bye identifier of on-card cryptographic material to be used in an authentication protocol to authenticate the credentials of a principal.

The reference data identifiers in the following table are allocated to the authentication of the listed principal.

Reference Data Identifier	Reference Data Name	Authenticated Principal	Security Status Type
'01'	Global PIN	Cardholder	Global
'02'-'09'	RFU	RFU	RFU
'0A'	ADM	Card Issuer	Global
'0B'-'0F'	RFU	RFU	RFU
'80'-'89'	RFU	RFU	RFU
'8A'-'8F'	Application Key	Application Provider	Application

4.14 Length

A length is an unsigned integer value between 0 and 2^{32}.

4.15 Offset

An offset is an unsigned integer value between 0 and 2^{16}.

4.16 Secure Channel Type

Encoding	Secure Channel Processing
'01'	Cryptographic checksum using symmetric key
‘02’	Digital signature using asymmetric key
‘03’	Encryption using symmetric key

4.17 Status Word

A status word is a 2-byte value returned by an application programming entry point or a card command. The first byte is referred to as SW1 and the second byte is referred to as SW2.

Recognized values of all SW1-SW2 pairs used as return values on both the client-application programming and card command interfaces and their interpretation are given in the following table. The description of individual client-application programming interface entry points or card commands may provide additional information for interpreting particular status words.

Table 4-7 – Status Words

SW1	SW2	Meaning
‘63’	‘00’	Meaning depends on the particular client-application programming interface entry point or card command returning it
‘63’	‘82’	End of data element encountered
‘68’	‘00’	Communication error
‘69’	‘82’	Security condition not satisfied
‘69’	‘83’	Authentication method blocked
‘69’	‘85’	Conditions of use not satisfied
‘69’	‘86’	Command not allowed, no current data element
‘69’	‘87’	Expected secure messaging data object missing
‘69’	‘88’	Incorrect secure messaging data objects
‘6A’	‘80’	Incorrect parameters in command data field
‘6A’	‘82’	Data element not found
‘6A’	‘86’	Incorrect parameters in P1 or P2
‘6A’	‘88’	Referenced data not found
‘6A’	‘89’	Data element already exists
'90'	'00'	Successful execution

5.0 CLIENT-APPLICATION PROGRAMMING INTERFACE

The following table lists all the entry points on the client-application programming interface and organizes them into three functional groups.

Entry Points for Communication	**Connect**
	Acquire Context
	Establish Secure Channel
	Release Context
	Disconnect
Entry Points for Card Application Management	**Add Card Application**
	Delete Card Application
	Generate Key Pair
	Import Key
	Get Card Application Properties
Entry Points for Data Management	**Create Data Element**
	Delete Data Element
	Select Data Element
	Get Data Element Properties
	Read Data
	Write Data
Entry Points for Authentication	**Authenticate Card**
	Authenticate Principal
	Get Certificate
	Get Challenge
	Create Digital Signature
	Verify Digital Signature

5.1 Entry Points for Communication

5.1.1 Connect

Purpose:	Connects the client-application programming interface and hence the client application itself to a specific PIV integrated circuit card.

Prototype:

```
status_word Connect(
   IN connection_description     connection,
   IN boolean                    sharedConnection,
   OUT handle                    cardHandle
);
```

Parameters:	**connection**	Description of a communication path from the platform on which the client-application is running to a specific PIV integrated circuit card.
	sharedConnection	If TRUE other client-applications can establish concurrent connections to the integrated circuit card. If FALSE and the connection is established then the calling client-application has exclusive access to the integrated circuit card.
	cardHandle	The returned opaque identifier of a communication channel to a particular integrated circuit card and hence of the card itself. cardHandle is used in all other entry points on the application programming interface to identify which card the operation of the entry point is to be applied.

Return Codes:

```
OK
CONNECTION_FAILURE
CONNECTION_LOCKED
```

5.1.2 Acquire Context

Purpose:	Establishes communication with and a context within a particular card application on the integrated circuit card. The context on acquisition typically includes a default dedicated file and a default transparent file in this dedicated file. The context will also include the results of processing the sequence of authenticators.

Prototype:

```
status_word AcquireContext(
   IN handle                     cardHandle,
   IN AID                        applicationAID,
   IN sequence of authenticator  authenticators
);
```

Parameters:	**cardHandle**	Identifier of the connected card containing the application to which the client-application wishes to be connected.

applicationAID	Identifier of the application to which the client-application desires to be connected.
authenticators	A sequence of zero or more authenticators to be used to authenticate the client-application to the card application and hence in establishing the initial security status in the card application context.

Return Codes:

```
OK
INVALID_CARD_HANDLE
APPLICATION_NOT_FOUND
AUTHENTICATION_FAILURE
```

5.1.3 Establish Secure Channel

Purpose: Create a secure channel from the application programming interface to the currently selected card application.

The secure channel construct appears on the client-application programming interface and describes cryptographic processing that is to be applied to data flowing between the client-application and a card application on the integrated circuit card.

The card command interface provides means to apply cryptographic processing to individual card commands. This means is called *secure messaging* and is described in Section 6.5 below.

When the integrated circuit card is in an interface device that is physically connected to the platform on which the client-application is running, a secure channel between the client-application and an application on the integrated circuit card can be implemented using secure messaging.

In the case that the client-application is remote from the interface device containing the integrated circuit card, additional cryptographic processing may be applied to the data flowing between the platform hosting the client-application and the interface device.

The description of a secure channel on the client-application programming interface applies only to the secure message processing that is to be applied to the card commands on the card command interface.

Where secure message processing is applied to card command interface commands and what other cryptographic processing might be applied to the data connection between the client-application programming interface and the card command interface is outside the scope of this chapter of this document.

The arguments to the Establish Secure Channel entry point below describe the cryptographic processing to be applied to the individual card commands that are sent to the integrated circuit card as a consequence of following calls to entry points on the client-application programming interface.

Prototype:

```
status_word EstablishSecureChannel(
   IN handle                     cardHandle,
   IN secure_channel_type        secureChannelType,
   IN reference_data_identifier  transmissionKey,
   IN reference_data_identifier  responseKey
);
```

Parameters:

cardHandle	Identifier of the card to which a secure channel is to be established.
secureChannelType	The type of the security to be placed on the channel.
transmissionKey	The reference data identifier on the integrated circuit card that is to be used to verify or decrypt the card command to which secure messaging processing has been applied.
responseKey	The reference data identifier on the integrated circuit card is to be used to apply secure message processing to the response to each card command.

Return Codes:

```
OK
INVALID_CARD_HANDLE
CHANNEL_ESTABLISH_FAILURE
```

5.1.4 Release Context

Purpose: Zeroize the context of the currently selected application including the application security state. The global security state is not affected. There is no currently selected application after successful return from this entry point.

Prototype:

```
status_word ReleaseContext(
   IN handle           cardHandle,
);
```

Parameters:

cardHandle	Identifier of the card to for which the context is to be released.

Return Codes:

```
OK
INVALID_CARD_HANDLE
NO_CURRENT_CONTEXT
```

5.1.5 Disconnect

Purpose: Perform a cold reset on the card and disconnect the application programming interface from the PIV and its command interface.

Prototype:

```
status_word Disconnect(
   IN handle            cardHandle,
);
```

Parameters:

cardHandle	Identifier of the card to be reset.

Return Codes:

```
OK
INVALID_CARD_HANDLE
```

5.2 Entry Points for Application Management

5.2.1 Add Card Application

Purpose: Adds a new card application to the integrated circuit card.

Prototype:

```
status_word AddCardApplication(
   IN AID                        applicationAID,
   IN application_description    applicationDescription,
   IN sequence of byte           applicationImplementation
);
```

Parameters:

applicationAID	The AID of the card application that is being added to the integrated circuit card.
applicationDescription	The description of the card application to be added to the integrated circuit card.
applicationImplementation	The implementation of the card application to be added to the integrated circuit card. The implementation can include data or executable code or both.

Return Codes:

```
OK
INVALID_CARD_HANDLE
SECURITY_CONDITIONS_NOT_SATISFIED
AID_EXISTS
INVALID_APPLICATION_DESCRIPTION
INVALID_APPLICATION_IMPLEMENTATION
```

5.2.2 Delete Card Application

Purpose: Deletes a card application from the integrated circuit card.

Prototype:

```
status_word DeleteCardApplication(
   IN handle            cardHandle,
   IN AID               applicationAID
);
```

Parameters:	**`cardHandle`**	Identifier of the connected card from which the application is to be deleted.
	`applicationAID`	The AID of the card application to be deleted from the integrated circuit card.

Return Codes: `OK`

```
INVALID_CARD_HANDLE
SECURITY_CONDITIONS_NOT_SATISFIED
APPLICATION_NOT_FOUND
```

5.2.3 Generate Asymmetric Key Pair

Purpose: Generate an asymmetric key pair in the currently selected card application.

Prototype:

```
status_word GenerateAsymmetricKeyPair(
   IN handle                     cardHandle,
   IN reference_data_identifier  keyIdentifier,
   IN algorithm_identifier       keyType,
   IN length                     keySize,
   OUT sequence of byte          publicExponent,
   OUT sequence of byte          publicKey
);
```

Parameters:	**`cardHandle`**	Identifier of the connected card on which the asymmetric key pair is to be generated.
	keyIdentifier	The reference_data_identifier that is to be associated to the generated key by the integrated circuit card.
	keyType	The description of the asymmetric key pair that is to be generated.
	keySize	The size in bits of the keys in the asymmetric key pair that is to be generated.
	publicExponent	The returned exponent of the generated asymmetric key pair.
	publicKey	The returned public key of the generated asymmetric key pair.

Return Codes:

```
OK
INVALID_CARD_HANDLE
SECURITY_CONDITIONS_NOT_SATISFIED
INVALID_ALGORITH_IDENTIFIER
INVALID_LENGTH
GENERATION_FAILURE
```

5.2.4 Import Key

Purpose: Loads a key into the currently selected card application.

Prototype:

```
status_word ImportKey(
   IN handle                      cardHandle,
   IN reference_data_identifier  keyIdentifier,
   IN sequence of byte           keyMaterial
);
```

Parameters:

cardHandle	Identifier of the connected card into one of whose card applications a key is to be loaded.
keyIdentifier	The reference_data_identifier that is to be associated to the generated key by the integrated circuit card.
keyMaterial	The key to be loaded into the card application.

Return Codes:

```
OK
INVALID_CARD_HANDLE
INVALID_KEY_FORMAT
```

5.2.5 Select Card Application

Purpose: Sets the currently selected card application.

Prototype:

```
status_word SelectCardApplication(
   IN handle                      cardHandle,
   IN AID                         applicationAID
);
```

Parameters:

cardHandle	Identifier of the connected card on which the currently selected application is to be selected.
applicationAID	The AID of the card application that is to become the currently selected card application.

Return Codes:

```
OK
INVALID_CARD_HANDLE
APPLICATION_NOT_FOUND
```

5.2.6 Get Card Application Properties

Purpose: Gets the properties of the currently selected card application.

Prototype:

```
status_word GetCardApplicationProperties (
   IN handle                      cardHandle,
   OUT application_properties    applicationProperties
);
```

Parameters:	**`cardHandle`**	Identifier of the connected card containing the application context whose properties are retrieved.
	`applicationProperties`	The returned properties of the currently selected application.

Return Codes:
```
OK
INVALID_CARD_HANDLE
```

5.3 Entry Points for Data Management

5.3.1 Create Data Element

Purpose: Creates a new data element or dedicated file in the file system of the currently selected card application. If successful, the new data element becomes the current data element in the currently selected card application or the dedicated file becomes the currently selected dedicated file in the currently selectedcard application. In the latter case the currently selected data element is undefined.

Prototype:
```
status_word CreateDataElement(
   IN handle                     cardHandle,
   IN data_element_name          dataElementName,
   IN data_element_properties    dataElementProperties,
   IN sequence of byte           dataElementContent
);
```

Parameters:	**`cardHandle`**	Identifier of the connected card in whose current application context a new data element is to be created.
	`dataElementName`	The name of the data element or dedicated file that is to be created.
	`dataElementProperties`	The properties of the new data element or dedicated file including the access control rules associated with the data element or dedicated file.
	`dataElementContent`	A sequence of bytes that become the initial content of the new data element if a data element is being created. Null if a dedicated file is being created.

Return Codes:
```
OK
INVALID_CARD_HANDLE
DATA_ELEMENT_NOT_FOUND
```

5.3.2 Delete Data Element

Purpose: Deletes a data element or dedicated file in the file system of the currently selected card application.

Prototype:

```
status_word DeleteDataElement(
   IN handle            cardHandle,
   IN data_element_name dataElementName
);
```

Parameters:

cardHandle	Identifier of the connected card from whose current card application a data element is to be deleted.
dataElementName	The name of the data element or dedicated file that is to be deleted.

Return Codes:

```
OK
INVALID_CARD_HANDLE
DATA_ELEMENT_NOT_FOUND
SECURITY_CONDITIONS_NOT_SATISFIED
```

5.3.3 Select Data Element

Purpose: Sets the currently selected data element in the currently selected application.

If the data elements in the currently selected application are hierarchically organized this entry point can also be used to select a dedicated file in the currently selected file system. This dedicated file becomes the currently selected dedicated file and the currently selected data element is undefined.

Prototype:

```
status_word SelectDataElement(
   IN handle            cardHandle,
   IN data_element_name dataElementName,
);
```

Parameters:

cardHandle	Identifier of the connected card in whose current application context a data element is to be selected and made the currently selected data element.
dataElementName	The name of the data element that is to become the current data element in the current application.

Return Codes:

```
OK
INVALID_CARD_HANDLE
DATA_ELEMENT_NOT_FOUND
```

5.3.4 Get Data Element Properties

Purpose:	Retrieves the properties of the currently selected dedicated file or data element in the currently selected application.

Prototype:

```
status_word GetDataElementProperties(
   IN handle            cardHandle,
   OUT sequence of byte dataElementProperties
);
```

Parameters:

cardHandle	Identifier of the connected card the properties of whose currently selected data elements are to be retrieved.
dataElementProperties	The returned properties of the currently selected data element.

Return Codes:

```
OK
INVALID_CARD_HANDLE
```

5.3.5 Read Data

Purpose:	Retrieve data from the currently selected data element.

Prototype:

```
status_word ReadData(
   IN handle            cardHandle,
   IN offset            offset,
   OUT sequence of byte data
);
```

Parameters:

cardHandle	Identifier of the connected card from whose currently selected data element data values are to be read and returned.
offset	Offset in bytes from the start of the data element at which reading is to being.
data	The returned data read from the data element starting at the provided offset.

Return Codes:

```
OK
INVALID_CARD_HANDLE
SECURITY_CONDITIONS_NOT_SATISFIED
NO_CURRENTLY_SELECTED_DATA_ELEMENT
```

5.3.6 Write Data

Purpose:	Updates the content of the currently selected data element.

Prototype:

```
status_word WriteData(
   IN handle            cardHandle,
   IN offset            offset,
   IN sequence of byte  data
);
```

Parameters:	**`cardHandle`**	Identifier of the connected card to whose currently selected data element data bytes are to be written.
	`offset`	Offset in bytes from the start of the data element at which writing is to being
	`data`	Data to be written to the data element.

Return Codes:
```
OK
INVALID_CARD_HANDLE
SECURITY_CONDITIONS_NOT_SATISFIED
NO_CURRENTLY_SELECTED_DATA_ELEMENT
```

5.4 Entry Points for Authentication

5.4.1 Authenticate Card

Purpose: Authenticate the integrated circuit card.

Prototype:
```
status_word AuthenticateCard(
   IN handle                     cardHandle,
   IN reference_data_identifier
                              referenceDataIdentifier,
   IN algorithm_identifier algorighmIdentifier,
   IN sequence of byte     challenge,
   OUT sequence of byte    authenticationData
);
```

Parameters:	**`cardHandle`**	Identifier of the connected card that is to be authenticated.
	referenceDataIdentifier	Identifier of the reference data to be used in the card authentication protocol.
	`algorithmIdentifier`	Identifier of the authentication algorithm to be used in the card authentication protocol.
	`challenge`	Random byte sequence to be used in the card authentication protocol.
	`authenticationData`	The returned cryptogram of the card authentication protocol.

Return Codes:
```
OK
INVALID_CARD_HANDLE
```

```
INVALID_REFERENCE_DATA_IDENTIFIER
INVALID_ALGORITHM_IDENTIFIER
```

5.4.2 Authenticate Principal

Purpose: Authenticate an identity to the integrated circuit card.

Prototype:

```
status_word AuthenticatePrincipal(
   IN handle                    cardHandle,
   IN reference_data_identifier
                        referenceDataIdentifier,
   IN algorithm_identifier algorithmIdentifier,
   IN sequence of byte     authenticationData,
);
```

Parameters:

cardHandle	Identifier of the connected card to which the principal is to be authenticated.
referenceDataIdentifier	Identifier of the reference data associated with the principal. The reference data identifier identifies the key or key set to be used in the authentication protocol.
algorighmIdentifier	Identifier of the authentication algorithm to be used in the authentication protocol.
authenticationData	Data to be used in the authentication protocol.

Return Codes:

```
OK
INVALID_CARD_HANDLE
INVALID_REFERENCE_DATA_IDENTIFIER
INVALID_ALGORITHM_IDENTIFIER
AUTHENTICATION_FAILURE
```

5.4.3 Get Certificate

Purpose: Retrieve a certificate from the integrated circuit card.

Prototype:

```
status_word GetCertificate(
   IN handle            cardHandle,
   IN sequence of byte  certificateIdentifier,
   OUT sequence of byte certificate
);
```

Parameters:

cardHandle	Identifier of the connected card from which a certificate is to be retrieved.
certificateIdentifier	Identifier of the certificate to be retrieved
certificate	The returned certificate.

Return Codes:
```
OK
INVALID_CARD_HANDLE
CERTIFICATE_NOT_FOUND
```

5.4.4 Get Challenge

Purpose: Retrieves a sequence of random bytes from the currently selected application.

Prototype:
```
status_word GetChallenge(
   IN handle               cardHandle,
   IN    length            length,
   OUT   sequence of byte  challenge
);
```

Parameters:

cardHandle	Identifier of the connected card to from which a random byte sequence is to be retrieved.
length	Number of random bytes to be returned.
challenge	Sequence of random bytes returned by the integrated circuit card.

Return Codes:
```
OK
INVALID_CARD_HANDLE
LENGTH_EXCEEDS_MAXIMUM
```

5.4.5 Create Digital Signature

Purpose: Creates digital signature.

Prototype:
```
status_word CreateDigitalSignature(
   IN handle                         cardHandle,
   IN reference_data_identifier      keyIdentifier,
   IN sequence of byte               messageToBeSigned,
   OUT sequence of byte              digitalSignature
);
```

Parameters:

cardHandle	Identifier of the connected card to from which a random byte sequence is to be retrieved.
keyIdentifier	Identifier of reference data to be used to create the digital signature.
messageToBeSigned	Sequence of bytes, for example a hash, for which a digital signature is to be created.
digitalSignature	The created digital signature.

Return Codes:
```
OK
```

```
INVALID_CARD_HANDLE
REFERENCE_DATA_NOT_FOUND
```

5.4.6 Verify Digital Signature

Purpose: Verifies a digital signature.

Prototype:

```
status_word VerifyDigitalSignature(
   IN handle                          cardHandle,
   IN reference_data_identifier       key,
   IN sequence of byte                signedMessage,
   IN sequence of byte                digitalSignature
);
```

Parameters:

`cardHandle`	Identifier of the connected card to from which a random byte sequence is to be retrieved.
`keyIdentifier`	Identifier of reference data to be used to verify the digital signature.
`signedMessage`	Sequence of bytes, for example a hash, that was signed.
`digitalSignature`	The digital signature on the signed message.

Return Codes:

```
OK
INVALID_CARD_HANDLE
INVALID_SIGNATURE
```

6. CARD PLATFORM COMMAND INTERFACE

The following table lists all the card commands on the card command platform interface and organizes them into four functional groups.

Type	Name	Secure Messaging	Command Chaining
Card Commands for Card Content Management	**CARD CONTENT MANAGEMENT REQUEST**	Yes	No
	LOAD	No	No
	DELETE APPLICATION	No	No
Card Platform Commands for Application Management	**GENERATE ASYMMETRIC KEY PAIR**	Yes	Yes
	SELECT APPLICATION	No	No
Card Platform Commands for Data Management	**CREATE FILE**	Yes	Yes
	DELETE FILE	No	No
	SELECT FILE	Yes	No
	GET DATA	Yes	No
	PUT DATA	Yes	Yes
	SEARCH BINARY	Yes	No
	READ BINARY	Yes	No
	UPDATE BINARY	Yes	Yes
Card Platform Commands for Authentication	**EXTERNAL AUTHENTICATE**	Yes	Yes
	GET CHALLENGE	No	No
	INTERNAL AUTHENTICATE	Yes	Yes
	VERIFY	Yes	No
	CHANGE REFERENCE DATA	Yes	No
	RESET RETRY COUNTER	Yes	No
	MANAGE SECURITY ENVIRONMENT	Yes	No
	PERFORM SECURITY OPERATION	Yes	Yes

6.1 Card Platform Commands for Card Content Management

6.1.1 CARD CONTENT MANAGEMENT REQUEST Command

The CARD CONTENT MANAGEMENT REQUEST (CCM REQUEST) sets the context on the integrated circuit card for loading a new application onto the card.

Command Syntax

CLA	‘00’ or ‘0C’
INS	‘yy’
P1	Request type
P2	‘00’
L_c	Length of the data field
Data Field	CCM REQUEST data
L_e	Absent

Table 6-1 - Coding of P1

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
-	-	1	0	0	0	0	0	Personalize an application
-	-	0	0	1	0	0	0	Make an application selectable
-	-	0	0	0	1	0	0	Install an application
-	-	0	0	0	0	1	0	Initiate loading of an application

Table 6-2 – Command Data Field for Personalization

Length (Bytes)	Description	M/O
2	‘0000’	M
1	Length of AID	M
5-16	AID	M
3	‘00000000’	M
1	Length of personalization data	O
Variable	Application-specific personalization data	O

Table 6-3 – Command Data Field for Make Selectable

Length (Bytes)	Description	M/O
2	‘0000’	M
1	Length of AID	M
5-16	AID	M
1	Length of application default security status	M
Variable	Application default security status	M
2	‘0000’	M

Table 6-4 – Command Data Field for Install

Length (Bytes)	Description	M/O
1	Length of AID of load file	M
5-16	AID of load file	M
1	Length of AID of executable module	M
5-16	AID of executable module	M
1	Length of AID of application when selectable	M
5-16	AID of application when selectable	M
1	Length of application default security status	M
Variable	Application default security status	M
1	Runtime environment installation parameters	M
Variable	Runtime environment installation parameters	M

Table 6-5 – Command Data Field for Load Initiation

Length (Bytes)	Description	M/O
1	‘EF’	C
1	Length of runtime environment parameters	C
Variable	Runtime environment parameters	C
1	‘C6’	C
1	‘02’	C
1	Persistent code space required by application	C
1	‘C7	C
1	‘02’	C
1	Volatile data space required by application	C
1	‘C8’	C
1	‘02’	C
2	Persistent data space required by application	C

Note: The command data field of the load initiation card content management request is a sequence of TLVs. Each TLV shall appear in its entirety or not at all.

Response Syntax

SW1	SW2	Meaning
'6A'	'80'	Incorrect parameters in the command data field.
'6A'	'86'	Incorrect parameters P1-P2
‘90’	‘00’	Successful execution

6.1.2 LOAD Command

The LOAD command transfers data blocks called *load file blocks* comprining a card application including executable code and data to the integrated circuit card.

Command Syntax

CLA	‘00’
INS	‘yy’
P1	'00' (more blocks) or ‘80’ (last block)
P2	Block number, ‘00’ to ‘FF’
L_c	Length of data field
Data Field	Load file block
L_e	Absent

Response Syntax

SW1	SW2	Meaning
‘69’	‘82’	Security condition not satisfied
‘90’	‘00’	Successful execution

6.1.3 DELETE APPLICATION Command

The DELETE APPLICATION command deletes a card application from the integrated circuit card.

Command Syntax

CLA	‘00’
INS	‘yy’
P1	'00'
P2	‘00’
L_c	Length of application identifier
Data Field	Application identifier
L_e	Absent

Response Syntax

SW1	SW2	Meaning
'69'	'82'	Security status not satisfied
'90'	'00'	Successful execution

6.2 Card Platform Commands for Application Management

6.2.1 GENERATE ASYMMETRIC KEY PAIR Command

The GENERATE ASYMMETRIC KEY PAIR command creates a new public/private key pair in the currently selected application on the integrated circuit card.

Command Syntax

CLA	As defined in Section 3.5
INS	'46'
P1	'02'
P2	'00' or reference data identifier of generated key
L_c	Length of the data field
Data Field	Sequence of data objects describing key pair to be generated
L_e	Absent or 'FF'

Description	**Tag**	**Status**
Cryptographic mechanism	'80'	Mandatory
Data element identifier of generated key material	'81'	Optional
Usage qualifier	'95'	Optional

Table 6-6 - Bytes of the Cryptographic Mechanism

Byte 1

b8	**b7**	**b6**	**b5**	**b4**	**b3**	**b2**	**b1**	**Meaning**
1	-	-	-	-	-	-	-	RSA-1024
-	1	-	-	-	-	-	-	RSA-2048
-	-	1	-	-	-	-	-	Etc.
-	-	-	1	-	-	-	-	
-	-	-	-	1	-	-	-	
-	-	-	-	-	1	-	-	
-	-	-	-	-	-	1	-	
-	-	-	-	-	-	-	1	

Table 6-7 - Bytes of the Usage Qualifier

Byte 1

b8	b7	b6	b5	b4	b3	b2	b1	Meaning
1	-	-	-	-	-	-	-	Signing
-	1	-	-	-	-	-	-	Encipherment
-	-	1	-	-	-	-	-	Etc.
-	-	-	1	-	-	-	-	
-	-	-	-	1	-	-	-	
-	-	-	-	-	1	-	-	
-	-	-	-	-	-	1	-	
-	-	-	-	-	-	-	1	

Response Syntax

Data Field	Public key data objects under template tag '7F49'
SW1-SW2	Status word

Description	Tag
OID of the cryptographic algorithm	'06'
RSA Public Key	
Modulus	'81'
Public exponent	'82'
DSA Public Key	'81'
First prime	'82'
Second prime	'83'
Basis	'84'
Public key	
ECDSA Public Key	
Prime	'81'
First coefficient	'82'
Second coefficient	'83'
Generator	'84'
Order	'85'
Public key	'86'

SW1	SW2	Meaning
'63'	'00'	Key pair generation failed – no further information
'90'	'00'	Successful execution

6.2.2 SELECT APPLICATION Command

The SELECT APPLICATION command sets the currently selected application.
If the currently selected application when the SELECT APPLICATION command is given is not the application whose AID is in the data field of the SELECT APPLICATION then the currently selected application is deselected and all of its current state is zeroized.
If the currently selected application when the SELECT APPLICATION command is given is the application whose AID is in the data field of the SELECT APPLICATION then all application security statuses are set to FALSE and the currently selected dedicated file is set to the application dedicated file of the application if any.

Command Syntax

CLA	'00'
INS	'A4'
P1	'04'
P2	'00' (no response data field) or '0C' (response data field is the selected application's properties)
L_c	Length of application identifier
Data Field	Application identifier
L_e	Absent

Response Syntax

Data Field	Application's properties when P2='0C'
SW1-SW2	Status word

SW1	SW2	Meaning
6A	82	Application not found
90	00	Successful execution

6.3 Card Platform Commands for Data Management

6.3.1 CREATE FILE Command

The CREATE FILE command creates a new dedicated or transparent file contained in the currently selected dedicated file of the currently selected application according to the file control parameter (FCP) given in the data field of the command.
The status word return '6A80' may indicate an improperly formatted FCP or an FCP that is missing mandatory data objects.
The length of the 'Number of data bytes in the file' data object is set to zero when creating a dedicated file.

Command Syntax

CLA	As defined in Section 3.5
INS	'E0'
P1	'00'
P2	'00'
L_c	Length of the file control parameter template in the data field
Data Field	File control parameter (FCP) template of the file to be created.
L_e	Absent

Table 6-8 - File Control Parameter Data Objects

Description	Tag	Length	Status
Number of data bytes in the file	'80'	Variable	Mandatory for TF
File descriptor byte	'82'	2	Mandatory
File identifier	'83'	2	Mandatory
Security attribute in expanded format	'AB'	Variable	Mandatory

Response Syntax

SW1	SW2	Meaning
'69'	'82'	Security status not satisfied
'6A'	'80'	Incorrect parameters in the command data field.
'6A'	'89'	Security status not satisfied
'90'	'00'	Successful execution

6.3.2 DELETE FILE Command

The DELETE FILE command deletes a dedicated or transparent file in the currently selected dedicated file of the currently selected application.

Command Syntax

CLA	'00'
INS	'E4'
P1	'00'
P2	'00'
L_c	2
Data Field	File identifier
L_e	Absent

Response Syntax

SW1	SW2	Meaning
'69'	'82'	Security status not satisfied
'6A'	'82'	File not found
'90'	'00'	Successful execution

6.3.3 SELECT FILE Command

The SELECT FILE command sets the currently selected dedicated file or the currently selected data element in the currently selected application.

Command Syntax

CLA	'00' or ‘0C’
INS	'A4'
P1	'00'
P2	'00' (no response) or ‘0C’ (response is properties of the selected entity)
L_c	2
Data Field	Dedicate file or data element identifier
L_e	Absent

Response Syntax

Data Field	Properties of the selected entity when P2=’0C’
SW1-SW2	Status word

SW1	SW2	Meaning
'6A'	'82'	File not found
90	00	Successful execution

6.3.4 GET DATA Command

The GET DATA command retrieves a BER-TLV data object.

If the currently selected file is a BER-TLV transparent file then the operation takes place within this file. Otherwise, the operation takes place in the currently selected dedicated file.

Command Syntax

CLA	‘00’ or ‘0C’
INS	'CA'
P1	High-order byte of a two-byte BER-TLV tag or '00'
P2	Low-order byte of a two-byte BER-TLV tag or one-byte BER-TLV tag if P1 is '00'
L_c	Absent
Data Field	Absent
L_e	Expected number of value bytes to be retrieved.

Response Syntax

Data Field	Value field of the BER-TLV data object identified by P1-P2
SW1-SW2	Status word

SW1	SW2	Meaning
'69'	'82'	Security status not satisfied
'6A'	'82'	Data object not found
'90'	'00'	Successful execution

6.3.5 PUT DATA Command

The PUT DATA command creates or updates a BER-TLV data object.

If the currently selected file is a BER-TLV transparent file then the operation takes place within this file. Otherwise, the operation takes place in the currently selected dedicated file.

If L_c is absent then the BER-TLV data object is deleted.

Command Syntax

CLA	As defined in Section 3.5
INS	'DA'
P1	High-order byte of a two-byte BER-TLV tag or '00'
P2	Low-order byte of a two-byte BER-TLV tag or one-byte BER-TLV tag if P1 is '00'
L_c	Length of data field
Data Field	Value of created or updated data object
L_e	Absent

Response Syntax

SW1	SW2	Meaning
'69'	'82'	Security status not satisfied
'90'	'00'	Successful execution

6.3.6 SEARCH BINARY Command

The SEARCH BINARY command is used search for the byte pattern in the data field of the command within the sequence of bytes comprising the content of the currently selected data element in the currently selected application.
The search is started in the content of the currently selected data element at the byte position given by the offset in the command. The search ends when the first sequence of bytes in the content are found that exactly match the sequence of bytes provided in the data field of the command or when no such sequence is found. In the former case, the byte position of the first byte of the matched content sequence is returned. In the latter case, an error condition is returned.

Command Syntax

CLA	'00' or ‘0C’
INS	'A0'
P1	High-order byte of 2-byte offset; high-order bit is 0.
P2	Low-order byte of 2-byte offset
L_c	Length of search pattern
Data Field	Search pattern
L_e	2

Response Syntax

Data Field	Byte position of the first byte in a sequence of Lc bytes at the provided offset or beyond that exactly matches the search pattern
SW1-SW2	Status word

SW1	SW2	Meaning
'62'	'82'	Pattern not found
'69'	'82'	Security status not satisfied
'69'	'86'	Card command not allowed (no current data element)
'90'	'00'	Successful execution

6.3.7 READ BINARY Command

The READ BINARY command is used to retrieve data from the currently selected data element in the currently selected application.
The data is retrieved from sequential byte positions in the data element starting at the byte position given by the offset in the command.
Data may be read beyond the end of the sequence of bytes that comprise the current content of the data element. The value of byte retrieved from a byte position that is beyond the end of the data currently in the data element is set to '00'. In this case a status word of '6381' is returned to indicate that reading has taken place beyond the end of the data currently in the data element.

Command Syntax

CLA	'00' or ‘0C’
INS	'B0'
P1	High-order byte of 2-byte offset; high-order bit is 0
P2	Low-order byte of 2-byte offset
L_c	Empty
Data Field	Empty
L_e	Number of bytes to be retrieved.

Response Syntax

Data Field	L_e bytes retrieved from currently selected data element
SW1-SW2	Status word

SW1	SW2	Meaning
'63'	'82'	Data retrieved beyond the end of the data element.
'69'	'82'	Security status not satisfied
'69'	'86'	Card command not allowed (no current data element)
'90'	'00'	Successful execution

6.3.8 UPDATE BINARY Command

The UPDATE BINARY command is used to write data into the currently selected data element in the currently selected application.
The data is written to sequential byte positions in the data element starting at the byte position given by the offset in the command.
Data may be written beyond the end of the sequence of bytes that comprise the current content of the data element. In this case the length of the content is set to the byte position of the last byte written plus one. In this case a status word of '6381' is returned to indicate that reading has taken place beyond the end of the data currently in the data element.

Command Syntax

CLA	As defined in Section 3.5
INS	'D6'
P1	High-order byte of 2-byte offset; high-order bit is 0.
P2	Low-order byte of 2-byte offset
L_c	Number of bytes to be written to the data element.
Data Field	Data to be written to the data element.
L_e	Empty

Response Syntax

SW1	SW2	Meaning
'63'	'81'	Data written beyond the end of the data element.
'69'	'82'	Security condition not satisfied
'69'	'86'	Command not allowed (no current data element)
'6A'	'84'	Insufficient storage area to write data.
'90'	'00'	Successful execution

6.4 Card Platform Commands for Authentication

6.4.1 EXTERNAL AUTHENTICATE Command

The EXTERNAL AUTHENTICATE command performs an authentication protocol using the authentication data provided in the data field of the command.

If the authentication protocol is successful, the security status of the principal identified in the P2 parameter is set to TRUE.

Command Syntax

CLA	As defined in Section 3.5
INS	'82'
P1	Algorithm identifier
P2	Reference data identifier
L_c	Length of data field
Data Field	Authentication data
L_e	Empty

Response Syntax

SW1	SW2	Meaning
'63'	'00'	Authentication failed
'6A'	'86'	Incorrect parameters P1-P2
'6A'	'88'	Referenced data not found
'90'	'00'	Successful execution

6.4.2 GET CHALLENGE Command

The GET CHALLENGE command returns a sequence of random bytes from the currently selected application. The returned sequence is retained in the currently selected application for possible use in a subsequent authentication protocol. If the currently selected application is deselected, the retained sequence is zeroized.

Command Syntax

CLA	'00'
INS	'84'
P1	'00'
P2	'00'
L_c	Empty
Data Field	Empty
L_e	Length in bytes of desired random byte sequence.

Response Syntax

Data Field	Sequence of L_e random bytes
SW1-SW2	Status word

SW1	SW2	Meaning
'61'	'xx'	'xx' additional bytes of information available
'90'	'00'	Successful execution

6.4.3 INTERNAL AUTHENTICATE Command

The INTERNAL AUTHENTICATE command performs an authentication protocol using the data provided in the data field of the command and returns the result of the authentication protocol in the response data field.

The INTERNAL AUTHENTICATE command is used to authenticate the card or a card application to the client-application.

Command Syntax

CLA	As defined in Section 3.5
INS	'88'
P1	Algorithm identifier
P2	'00'
L_c	Length of data field
Data Field	Challenge
L_e	Length of expected cryptogram

Response Syntax

SW1	SW2	Meaning
'6A'	'86'	Incorrect P1 parameter
'90'	'00'	Successful execution

6.4.4 VERIFY Command

The VERIFY command initiates the comparison in the card of stored reference data with authentication data in the data field of the command.

Command Syntax

CLA	'00' or ‘0C’
INS	'20'
P1	'00'
P2	Reference data identifier
L_c	Length of data field
Data Field	Authentication data (i.e., password or PIN)
L_e	Empty

Response Syntax

SW1	SW2	Meaning
'63'	'00'	Verification failed
'63'	'CX'	Verification failed, X indicates the number of further allowed retries
'69'	'83'	Authentication method blocked
'6A'	'88'	Reference data not found
'90'	'00'	Successful execution

6.4.5 CHANGE REFERENCE DATA Command

The CHANGE REFERENCE DATA replaces reference data stored on the card with new reference data after comparing authentication data with the current reference data.

Command Syntax

CLA	'00' or ‘0C’
INS	'20'
P1	'00'
P2	Reference data identifier
L_c	Length of data field
Data Field	Authentication data (i.e., password or PIN) followed by new reference data
L_e	Empty

Response Syntax

SW1	SW2	Meaning
'63'	'00'	Verification failed
'63'	'CX'	Verification failed, X indicates the number of further allowed retries
'69'	'83'	Authentication method blocked
'6A'	'88'	Reference data not found
'90'	'00'	Successful execution

6.4.6 RESET RETRY COUNTER Command

The RESET RETRY COUNTER resets the reference data retry counter of the identified reference data to its initial value.

The data field of the command contains the resetting code, for example an unblocking PIN. If the resetting code is to be authenticated, then P2 contains the reference data identifier of the data to be used for authentication. Otherwise, P2 is set to ‘00’.

Command Syntax

CLA	'00' or ‘0C’
INS	'2C'
P1	'01'
P2	‘00’ or reference data identifier
L_c	Length of data field
Data Field	Resetting code.
L_e	Empty

Response Syntax

SW1	SW2	Meaning
'63'	'00'	Verification of resetting code failed
'6A'	'88'	Reference data not found
90	00	Successful execution

6.4.7 MANAGE SECURITY ENVIRONMENT Command

The MANAGE SECURITY ENVIRONMENT command provides parameters to subsequent PERFORM SECURITY OPERATION commands.

Command Syntax

CLA	'00' or ‘0C’
INS	'22'
P1	'01'
P2	'B6'
L_c	Length of the data field
Data Field	Data objects to be placed in the control reference template for digital signature ('B6') in the current security environment
L_e	Empty

Table 6-9 – MANAGE SECURITY ENVIRONMENT Data Objects

Description	Tag
Cryptographic mechanism reference	'80'
File reference	'81'
AID	'82'
Reference data index of private key	'84'

Response Syntax

SW1	SW2	Meaning
'6A'	'80'	Invalid or missing tag, length or value in a data object in the command data field
'6A'	'86'	Incorrect parameters P1-P2
'90'	'00'	Successful execution

6.4.8 PERFORM SECURITY OPERATION Command

The PERFORM SECURITY OPERTION command performs a digital signature computation or verification according to the parameters and data in the current security environment.

Command Syntax

CLA	As defined in Section 3.5
INS	'2A'
P1	See below
P2	See below
L_c	Absent or length in bytes of command data field
Data Field	Absent, hash-value data object ('90') or digital signature data object ('9E')
L_e	Length of expected response

Table 6-10 – Perform Security Operation Command Parameters and Data Objects

P1	P2	Tag	Digital Signature Operation
'9E'	'9A'	'90'	The value field of the data object in command data field is signed and the signature returned as the response to the command
'90'	'A8'	'90'	The value field of the data object in the command data field is the hash code the signature on which is to be verified by a subsequent digital signature verification operation.
'00'	'A8'	'9E'	The value field of the data object in the command data field is the signature on the hash-code in the current security environment and this signature is verified.

Response Syntax

Data Field	Digital signature when P2 is '9A' otherwise absent.
SW1-SW2	Status word

SW1	SW2	Meaning
'63'	'00'	Verification failed
'69'	'85'	Insufficient information in current security environment.
'6A'	'80'	Incorrect parameters in command data field
'6A'	'86'	Incorrect parameters P1-P2
'90'	'00'	Successful execution

6.5 Secure Messaging

The application of secure message processing to a card command is indicated by setting the class byte (CLA) of the command to ‘0C’.

The data field of a card command to which secure message processing has been applied consists of a sequence of BER-TLV data objects one of which shall be one of the control reference templates in Table xx below.

Table 6-11 – Secure Messaging Control Reference Templates

Description	Tag	M/O
Control reference template for cryptographic checksum	‘B4’	C
Control reference template for digital signature	‘B6’	C
Control reference template for confidentiality	‘B8’	C

The four bytes of the command header are included in the computation of the cryptographic checksum and the digital signature. The bytes comprising the secure message data objects are not included in the computation of the cryptographic checksum or digital signature.

If a response descriptor template appears in the data field of a card command then secure message processing shall be applied to the data field and status word of the response to the card command.

6.5.1 Cryptographic Checksum

The following data objects appear in the command data field of a card command to which secure messaging using cryptographic checksum has been applied.

Table 6-12 – Data Objects in the Command Field of Secure Messaging with Cryptographic Checksum

Description	Tag	Comment	M/O
Plain value not encoded in BER-TLV	‘80’	Data field of the card command	M
Control reference template for cryptographic check sum	‘B4’	Describes the processing used to c0mpute the cryptographic checksum	M
Cryptographic checksum	'8E'	Cryptographic checksum of the 4-byte header and the command data field	M
Reponse descriptor template	‘BA’	Description of the secure message processing that shall be applied to the response	O

The following data objects appear in the control reference template for cryptographic checksum.

Table 6-13 – Data Objects in the Control Reference Template for Cryptographic Checksum

Description	Tag	Comment	M/O
Reference data identifier	‘83’	Identifier of the secret key used to compute the cryptographic checksum	M

6.5.2 Digital Signature

The following data objects appear in the command data field of a card command to which secure messaging using digital signature has been applied.

Table 6-14 – Data Objects in the Command Field of Secure Messaging with Digital Signature

Description	Tag	Comment	M/O
Plain value not encoded in BER-TLV	'80'	Data field of the card command	M
Control reference template for digital signature	‘B6’	Describes the processing used to compute the digital signature	M
Digital signature	'9E'	Digital signature on the 4-byte header and the command data field	M
Reponse descriptor template	‘BA’	Description of the secure message processing that shall be applied to the response	O

The following data objects appear in the control reference template for digital signature.

Table 6-15 – Data Objects in the Control Reference Template for Digital Signature

Description	Tag	Comment	M/O
Reference data identifier of public key	‘83’	Public key to use to verify the digital signature	C
Reference data identifier of private key	‘84’	Private key to use to verify the digital signature	C

Note: Either the reference data identifier of a public key or the reference data indicator of a private key must appear in the control reference template for digital signature.

6.5.3 Confidentiality

The following data objects appear in the command data field of a card command to which secure messaging using confidentiality has been applied.

Table 6-16 – Data Objects in the Command Field of Secure Messaging with Confidentiality

Description	Tag	Comment	M/O
Padding-content indicator byte followed by cryptogram of plain value not encoded in BER-TLV	'86'	Encryption of the data field of the card command preceeded by a padding-content indicator byte	M
Control reference template for confidentiality	‘B6’	Describes the processing used to compute the cryptogram of the command data field	M
Reponse descriptor template	‘BA’	Description of the secure message processing that shall be applied to the response	O

The following data objects appear in the control reference template for confidentiality.

Table 6-17 – Data Objects in the Control Reference Template for Confidentiality

Description	Tag	Comment	M/O
Reference data identifier	‘83’	Identifier of the secret key used to encrypt the command data field	M

6.5.4 Response

The data objects in the response descriptor template in the data field of a card command to which secure message processing has been applied describe secure message processing that is to be applied by the integrated circuit card to the response to the card command.

The following data objects appear in the response descriptor template.

Table 6-18 – Data Objects in the Respose Descriptor Template

Description	Tag	Comment	M/O
Reference data identifier	‘83’	Identifier of the secret key used to compute the cryptographic checksum of the response and the status word	M

The following objects appear in the response data field to which secure message processing has been applied.

Table 6-19 – Data Objects in the Response Data Field using Secure Messaging

Description	Tag	Comment	M/O
Plain value not encoded in BER-TLV	'80'	Response data field	M
Processing status	'99'	Status word generated by the card command	M
Cryptographic checksum	'8E'	Cryptographic checksum of the response data field and the status word	M

6.6 Command Chaining

ISO/IEC 7816-4 provides command chaining for two purposes: for the transmission of a command data field too long for a single command and for multi-step transaction processing. The class (CLA) byte in a sequence of chained commands is constant except for bit 5. Bit 5 is set to 1 for all commands in the chain except the last command and is set to 0 to indicate that the command is the last (or only) command in the sequence.

In the case that command chaining is used for the transmission of a command data field too long for a single command, the INS, P1 and P2 command header bytes shall be constant across all commands in the chain. The syntax and semantics of these common INS, P1 and P2 command header bytes are the same in both chained and unchained use. That is, the card command defined by the chain of card commands is the common header together a command data field that is the concatenation of all of the command data fields in the chain. All commands in the present document that support command chaining use of type of command chaining.

In the case that command chaining is used for a multi-step trasaction, the INS, P1 and P2 command header bytes are not constant across all commands in the chain. The semantics of the INS, P1 and P2 command header bytes within the commands in a chain used for a multi-step process is command-specific and is described as part of the description of the commands involved. If no such description is provided then the command can be only be chained for the purpose of transmitting a command data field that is too long for a single command. No card commands in the current document support this type of command chaining.

6.7 Cryptographic Information Application

The *cryptographic information application* is a codified and structured database of information about the cryptographic capabilities on the PIV integrated circuit card as described generally in ISO/IEC 7816-15.

The AID of the cryptographic information application on the PIV integrated circuit card is 'E8 28 BD 08 0F 00'. Selecting this AID sets the currently selected dedicated file to the ADF DF.CIA as described in ISO/IEC 7816-15. The transparent files and dedicated files found in DF.CIA are described below.

The inclusion of the cryptographic information application on a PIV integrated circuit card is optional but if it is present it has the below contents and structure.

EF.CardInfo

EF.CardInfo is mandatory and it shall contain only version number of the Cryptographic Information Application. The file identifier of EF.CardInfo is ‘5032’.

```
CardInfo ::= SEQUENCE { version INTEGER {v1(0),v2(1)} (v1|v2,...)}
```

EF.OD

EF.OD is mandatory and it shall contain only paths to the files listed below. The file identifier of EF.CardInfo is ‘5031’.

```
CIOChoice ::= CHOICE {
    privateKeys  [0] PrivateKeys,
    publicKeys   [1] PublicKeys,
    secretKeys   [3] SecretKeys,
    certificates [4] Certificates,
    authObjects  [8] AuthObjects
}

PrivateKeys          ::= PathOrObjects {PrivateKeyChoice}
PublicKeys           ::= PathOrObjects {PublicKeyChoice}
SecretKeys           ::= PathOrObjects {SecretKeyChoice}
Certificates         ::= PathOrObjects {CertificateChoice}
AuthObjects          ::= PathOrObjects {AuthenticationObjectChoice}

PathOrObjects {ObjectType} ::= CHOICE {
    path Path
}
```

EF.PrKD

The private key description file shall contain the following objects for each private key:

```
CommonObjectAttributes ::= SEQUENCE {
    label Label OPTIONAL
}

CommonKeyAttributes ::= SEQUENCE {
    iD Identifier,
    usage KeyUsageFlags,
    keyReference KeyReference MANDATORY
   }

CommonPrivateKeyAttributes ::= SEQUENCE {
    name Name OPTIONAL,
    keyIdentifiers [0] SEQUENCE OF CredentialIdentifier {{KeyIdentifiers}} OPTIONAL
   }
```

EF.PuKD

The public key description file shall contain the following objects for each public key

```
CommonObjectAttributes ::= SEQUENCE {
    label Label OPTIONAL
}

CommonKeyAttributes ::= SEQUENCE {
    iD Identifier,
    usage KeyUsageFlags,
    keyReference KeyReference MANDATORY
   }

CommonPublicKeyAttributes ::= SEQUENCE {
    name Name OPTIONAL,
    keyIdentifiers [0] SEQUENCE OF CredentialIdentifier {{KeyIdentifiers}} OPTIONAL
   }
```

EF.SKD

The secret key description file shall contain the following objects for each secret key.

```
CommonObjectAttributes ::= SEQUENCE {
    label Label OPTIONAL
}

CommonKeyAttributes ::= SEQUENCE {
    iD Identifier,
    usage KeyUsageFlags,
    keyReference KeyReference MANDATORY
   }

CommonSecretKeyAttributes ::= SEQUENCE {
    keyLen INTEGER MANDATORY, -- keylength (in bits)
}
```

EF.CD

The certificate description file shall contain the following objects for each certificate.

```
CommonObjectAttributes ::= SEQUENCE {
    label Label OPTIONAL
}

CommonCertificateAttributes ::= SEQUENCE {
    iD Identifier,
    identifier CredentialIdentifier {{KeyIdentifiers}} OPTIONAL,
    trustedUsage [1] Usage OPTIONAL,
    identifiers [2] SEQUENCE OF CredentialIdentifier
   }
```

EF.AOD

The authentication object description file shall contain the following objects for each authentication object.

```
CommonObjectAttributes ::= SEQUENCE {
    label Label OPTIONAL
}

CommonAuthenticationObjectAttributes ::= SEQUENCE {
    authId Identifier OPTIONAL,
    authReference Reference OPTIONAL,
    seIdentifier [0] Reference OPTIONAL
}
```

7. COMMON DATA MODEL FOR PERSONAL IDENTIFICATION VERIFICATION

7.1 Overview

The FIPS 201 Common Data Model for Personal Identification Verification is TBD.

8. REFERENCES

8.1 ETSI

ETSI TS 102 221, *Smart cards; UICC-Terminal interface; Physical and logical characteristics*

8.2 FIPS

FIPS 140-2, *Security Requirements for Cryptographic Modules*, June, 2001

FIPS 180-2, *Secure Hash Standard (SHS)*, August, 2002.

FIPS 186-2, *Digital Signature Standard (DSS)*, January, 2000

FIPS 190, *Guideline for the Use of Advanced Authentication Technology Alternatives*, September, 1994

FIPS 196, *Entity Authentication Using Public Key Cryptography*, February, 1997

FIPS 197, *Advanced Encryption Standard*, November, 2001

FIPS 198, *The Keyed-Hash Message Authentication Code (HMAC)*, March, 2002

8.3 IETF

IETF RFC 1738:1994, *Uniform resource locators (URL)*

IETF RFC 1778:1995, *The String Representation of Standard Attribute Syntaxes*

IETF RFC 2396:1998, *Uniform resource locators (URL): General syntax*

8.4 ISO/IEC

ISO 3166-1:1997, *Codes for the representation of names of countries and their subdivisions — Part 1: Country codes*

ISO 7498-2:1989 Information Processing Systems - Open Systems Interconnection - Reference Model - Part 2: Security Architecture

ISO/IEC 7812-1:2000, *Identification cards — Identification of issuers — Part 1: Numbering system*

ISO/IEC 7816 (all parts), *Information technology — Identification cards — Integrated circuit(s) cards with contacts*

ISO/IEC 8824-2:2002, *Information technology -- Abstract Syntax Notation One (ASN.1): Information object specification*

ISO/IEC 8825-1:2002, *Information technology — ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)*

ISO/IEC 9796 (all parts), *Information technology — Security techniques — Digital signature schemes giving message recovery*

ISO/IEC 9797 (all parts), *Information technology — Security techniques — Message authentication codes (MACs)*

ISO/IEC 9798 (all parts), *Information technology — Security techniques — Entity authentication*

ISO/IEC 9979:1999, *Information technology — Security techniques — Procedures for the registration of cryptographic algorithms*

ISO 9992-2:1998, *Financial transaction cards — Messages between the integrated circuit card and the card accepting device — Part 2: Functions, messages (commands and responses), data elements and structures*

ISO/IEC 10116:1997, *Information technology — Security techniques — Modes of operation for an n-bit block cipher*

ISO/IEC 10118 (all parts), *Information technology — Security techniques — Hash-functions*

ISO/IEC 10536 (all parts), *Information technology — Identification cards – Contactless integrated circuit(s) cards – Close-coupled cards*

ISO/IEC 11770 (all parts), *Information technology — Security techniques — Key management*

ISO/IEC 14443 (all parts), *Information technology — Identification cards – Contactless integrated*
circuit(s) cards – Proximity cards

ISO/IEC 14888 (all parts), *Information technology — Security techniques — Digital signatures with appendix*

ISO/IEC 15693 (all parts), *Information technology — Identification cards – Contactless integrated circuit(s) cards – Vicinity cards*

ISO/IEC 18033 (all parts), *Information technology — Security techniques — Encryption algorithms*

ISO/IEC 24727-1 *Identification cards — Integrated circuit(s) cards programming interfaces — Architecture.*

ISO/IEC 24727-2 *Identification cards — Integrated circuit(s) cards programming interfaces — Generic card edge.*

ISO/IEC 24727-3 *Identification cards — Integrated circuit(s) cards programming interfaces — Programming interface.*

8.5 ITU

ITU X.680 (07/02), *Information technology - Abstract Syntax Notation One (ASN.1): Specification of basic notation*

ITU X.681 (07/02), *Information technology - Abstract Syntax Notation One (ASN.1): Information object specification*